IN THE CLAIMS:

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (Currently Amended) A computer-implemented method for dynamic data type enrichment comprising ~~the steps~~:

loading an application program into memory, the application program comprising a variable that is defined as an instance of both a basic data type and a specific data type;

accessing metadata at runtime to map the variable to a definition of the specific type;

dynamically adding the basic data type of the variable with metadata at runtime; and

processing the variable consistently with the metadata definition of the specific data type;

wherein the application program automatically processes changes in the metadata without incurring changes in a computer system implementing the method.

2. (Previously Presented) The method of claim 1, wherein the application program uses an application programming interface for accessing the metadata.

3. (Previously Presented) The method of claim 2, wherein the application program calls through the application programming interface at least one metadata service that relates to the basic data type.

4. (Previously Presented) The method of claim 3, wherein the at least one metadata service copies the metadata to a metadata cache.

5. (Previously Presented) The method of claim 1, wherein the basic data type is defined in one ore more of Visual Basic, Java, or C++.

6. (Previously Presented) The method of claim 1, wherein the metadata defines an allowed value range for the specific data type that is a subset of an allowed value range for the basic data type.

7. (Previously Presented) The method of claim 1, wherein the metadata defines a text label for a user input field.

8. (Previously Presented) The method of claim 7, wherein the variable is set to a value entered into the user input field.

9. (Currently Amended) The method of claim 6, wherein the metadata is stored in a private instance of the metadata store together with the application program such that a user can perform actions leading to changes in the metadata without impacting other users of the application.

10. (Currently Amended) The method of claim 6, wherein the metadata is stored in a shared instance of the metadata store such that a user can introduce changes in the metadata that affect all users of the application program.

11. (Cancelled).

12. (Currently Amended) A computer system comprising:

a memory storing an application program that comprises a variable that is defined as an instance of both a basic data type and a specific data type; and

a processor executing instructions to:

access metadata at runtime to map the variable to a definition of the specific data type,

dynamically adding the basic data type of the variable with metadata at runtime; and

process the variable consistently with the metadata definition of the specific data type;

wherein the application program automatically processes changes in the metadata without incurring changes in a computer system.

13. (Previously Presented) The computer system of claim 12 further comprising an application programming interface to access the metadata from the application program.

14. (Previously Presented) The computer system of claim 13, wherein the application programming interface provides at least one metadata service that relates to the basic data type used by the application program.

15. (Previously Presented) The computer system of claim 14, further comprising a metadata cache, the at least one metadata service copying the metadata to the metadata cache.

16. (Previously Presented) The computer system of claim 12, wherein the basic data type is defined in one or more of Visual Basic, Java, or C++.

17. (Previously Presented) The computer system of claim 16, wherein the metadata defines an allowed value range for the specific data type that is a subset of an allowed value range for the basic data type.

18. (Previously Presented) The computer system of claim 17, wherein the metadata defines a text label for a user input field.

19. (Previously Presented) The computer system of claim 18, wherein the variable is set to a value entered into the user input field.

20. (Currently Amended) The computer system of claim 17, wherein the metadata is stored in a private instance of the metadata store together with the application program such that a user can perform actions leading to changes in the metadata without impacting other users of the application.

21. (Currently Amended) The computer system of claim 17, wherein the metadata is stored in a shared instance of the metadata store, such that a user can introduce changes in the metadata that affect all users of the application program.

22. (Withdrawn) A method for generating an application program (210) comprising the steps:

making available at least one metadata service (191) to be used in the application program (210) at design time for defining how the application program (210) can access metadata (150) at runtime; and

including a first implementation portion of the at least one metadata service (191) in the IDE (800) that is unaffected by changes of a second implementation portion of the least one metadata service (191) in a metadata store (220).

23. (Withdrawn) An integrated development environment (IDE) (800) for generating an application program (210) by performing the steps of claim 22.

24. (Cancelled).

25. (Previously Presented) The method according to claim 1, wherein the variable is declared as an instance of both the basic data type and the specific data type at design time.

26. (Currently Amended) A computer-implemented method for dynamic dta type enrichment comprising:

loading an application program into memory, the application program comprising a variable that is defined as an instance of both a basic data type and a specific data type;

accessing metadata over a network at runtime using an application programming interface, the metadata mapping the variable to a definition of the specific data type that indicates a text label for an input field to the variable and indicating allowed value range for the variable;

dynamically adding the basic data type of the variable with metadata at runtime; and

processing the variable consistently with the indicated allowed value range;

wherein the application program automatically processes changes in the metadata without incurring changes in a computer system implementing the method.

27. (Currently Amended) A computer program product comprising instructions embodied on a memory of a computer system that cause at least one processor of the computer system to execute a method, the method comprising:

loading an application program into memory, the application program comprising a variable that is defined as an instance of both a basic data type and a specific data type;

accessing metadata at runtime to map the variable to a definition of the specific data type;

dynamically adding the basic data type of the variable with metadata at runtime; and

processing the variable consistently with the metadata definition of the specific data type;

wherein the application program automatically processes changes in the metadata without incurring changes in a computer system.

28. (Previously Presented) The computer program product according to claim 27, wherein the metadata is stored on a hard disk, and the application program loaded into memory accesses the metadata by retrieving the metadata from the hard disk.

29. (New) The method of claim 1, wherein the metadata is stored in a backend system such that the application program executes on the backend system and the user interface is assembled in a different system that is separate from the computer system.

30. (New) The method of claim 29, wherein the different system is a portal system using a portal runtime framework.

31. (New) The method of claim 1. wherein the metadata is stored together with other metadata in a metadata store, and the metadata store is unknown to an application developer at design time.

32. (New) The method of claim 31, further comprising providing metadata to the application program before and after changing the metadata store at runtime of the application program without causing a system failure.

33. (New) The method of claim 32, further comprising changing a second implementation portion of metadata services on a side of the metadata store without changing a first implementation portion of metadata services on a side of an integrated development environment.

34. (New) The computer system of claim 12, wherein the metadata is stored in a backend system such that the application program executes on the backend system and the user interface is assembled in a different system that is separate from the computer system.

35. (New) The computer system of claim 34, wherein the different system is a portal system, using a portal runtime framework.

36. (New) The computer system of claim 12, wherein the metadata is stored together with other metadata in a metadata store, and the metadata store is unknown to an application developer at design time.

37. (New) The computer system of claim 36, wherein the processor further executes instructions to provide metadata to the application program before and after changing the metadata store at runtime of the application program without causing a system failure.

38. (New) The computer system of claim 37, wherein the processor further executes instructions to change a second implementation portion of metadata services on a

side of the metadata store without changing a first implementation portion of metadata services on a side of an integrated development environment.